NUS | Computing
National University of Singapore

**CS3230**
eric_han@nus.edu.sg
https://eric-han.com
*Computer Science*

T06 – Week 7

# Randomized Algorithms

*CS3230 – Design and Analysis of Algorithms*

# Admin

## Assignment 3

### Q1 - NUS OSA cards

- Be aware that the focus is on **query complexity**, not just time complexity. Some answers described time complexity, which is fine here but not always accurate.
- **Use MT** when possible. Some used other techniques, which are okay, but MT is the preferred lazy approach.
- Ensure you **write down the recurrence relation** when using a recursive algorithm before discussing it.

### Q2 - Inversions in array

- Avoid **brute force** approaches—consider faster algorithms.
- Otherwise, most solutions look fine.

### Q3 - Bubble sort

- No major issues noted.

## Assignment 4

### Q1 - Two largest numbers

- Some **missed discussing why** the answer involves **n**. Show some working (e.g., **(n-2) + 1**) or similar reasoning.

### Q2 - Counting sort

- Clearly **identify which lines use k vs. n complexity** and summarize the conclusion accordingly.

### Q3 Heavier Balls

- Many struggled with this question.
  **Tip: Break it down into cases**—using a table may help with analysis.
- Generally **poorly attempted**; if incomplete, study the solution and try to reattempt it.

# Mid-semester Survey

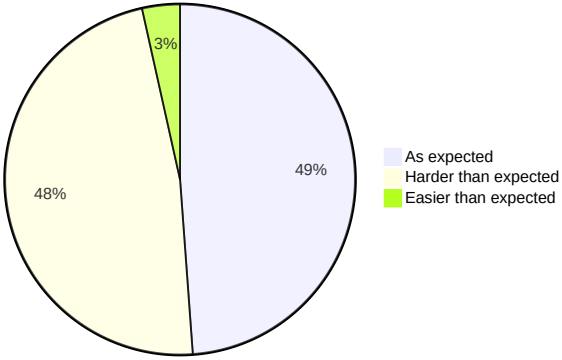


Figure 1: How do you find the difficulty of the course? [Module]

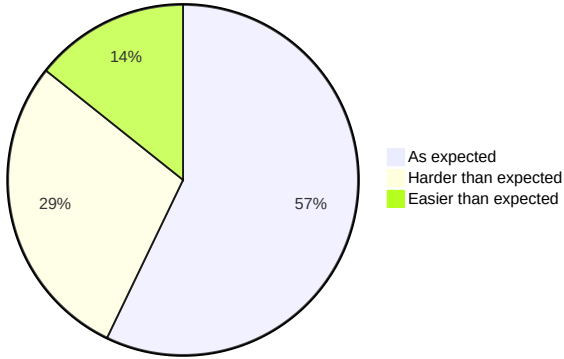


Figure 2: How do you find the difficulty of the course? [TG19]

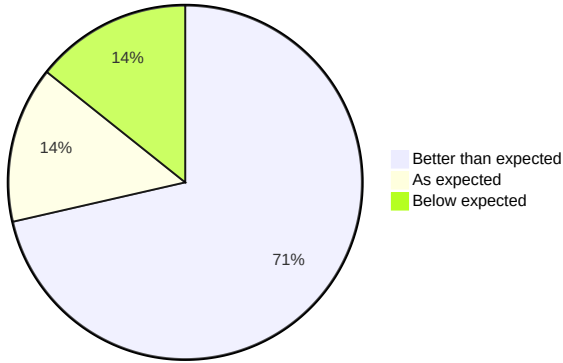


Figure 3: Your assessment of your TG tutor, based on how he/she runs the tutorial [TG19]

**Please let us know any comments/suggestions about the course so far. [TG19]**

- **Tutorial Participation:**
  While "participation is good for learning and interactivity," the emphasis on it may be too strong. The student feels they "can only learn so much from other students answering the questions" and suggests placing "more emphasis on teaching the algorithms and concepts for better learning."
- **Mathematical Proofs vs. Algorithms:**
  The student prefers "less emphasis on full-on mathematical proofs like in assignment 1, q1b" and would rather see "more questions instead on algorithms." If proofs are included, they suggest clarifying "how this ties in to the overall goal of the course."
- **Lecture Pace:**
  Solutions in lectures are "covered too quickly," making "the reasonings behind the proofs more abstract and difficult to understand." They recommend a "slower pace to ensure that the reasonings and concepts can be effectively conveyed."
- **Appreciation:**
  The student acknowledges that these concerns are not due to the tutor and expresses gratitude for the "teaching team's dedication" and for holding a mid-semester survey.

**Responses**

- **Tutorial Participation:**[1] *(Need to balance participation vs engaging with content taught in lectures*[2]*) If you have a specific suggestion here, please come talk to me.*
  - Attendance: Just be present.
  - Participation x2: Our class often does not require students to present full solutions, and are often just discuss intuition and I present the detailed answer.
- **Mathematical Proofs vs. Algorithms:** These math proofs often appear during the analysis for example A1 Q1B appears in randomized quicksort analysis. We will consider putting these clarifications in the next iteration of CS3230.
- **Lecture Pace:** We will keep it not too fast and explain the intuition if possible.
- **Appreciation:** Thanks!

Thanks for your professional feedback, lets improve together!

[1] Submit these under module feedback, not teaching feedback, in the end-of-sem review.

[2] Teaching the algorithms is done during lectures.

# Lecture Review

**Techniques**

- Linearity of expectations
- Indicator random variables
- Markov inequality
- Union bound
- Principle of deferred decision
- Amplification of success probability

**Algorithms**

- Freivalds' algorithm (Monte Carlo)
- (Randomized) Quick Sort (Las Vegas)

**Balls and Bins**

- Coupon collector (probability of no empty bin)
- Chain hashing (expected bin size)

# Question 1 [G]/[P1]

In the class, we showed that Freivalds' algorithm succeeds with a probability of at least $1/2$.

Show that the bound $1/2$ in the analysis is actually **the best possible** by constructing an input $(A, B, C)$ on which the success probability of Freivalds' algorithm is precisely $1/2$.

**Answer**

**Best Possible $1 \times 1$ Matrix Example**

Consider $1 \times 1$ matrices: $A = (1), B = (0), C = (1) \implies AB = (1)(0) = 0 \neq C$.
In Freivalds' algorithm, let $v = (v_1)$, where $v_1$ is randomly chosen from $\{0, 1\}$:

- **Incorrect** with probability $1/2$, $v_1 = 0$, so $A(Bv) = Cv$.
- **Correct** with probability $1/2$, $v_1 = 1$, so $A(Bv) \neq Cv$.

Thus, the probability of success is exactly $1/2$.

**Answer**

**Best Possible $1 \times 1$ Matrix Example**

Consider $1 \times 1$ matrices: $A = (1), B = (0), C = (1) \implies AB = (1)(0) = 0 \neq C$.
In Freivalds' algorithm, let $v = (v_1)$, where $v_1$ is randomly chosen from $\{0, 1\}$:

- **Incorrect** with probability $1/2$, $v_1 = 0$, so $A(Bv) = Cv$.
- **Correct** with probability $1/2$, $v_1 = 1$, so $A(Bv) \neq Cv$.

Thus, the probability of success is exactly $1/2$.

**Generalizing to $n \times n$ Matrices**

Appending zeros extends this construction to any $n \times n$ matrix. For $n = 3$, we use:

$$A = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad B = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}, \quad C = \begin{pmatrix} 1 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \end{pmatrix}$$

and we still have: $A(Bv) = Cv$ if $v_1 = 0$ and $A(Bv) \neq Cv$ if $v_1 = 1$.

# Equality Testing Problem

Alice holds an $n$-bit string $S_A \in \{0,1\}^n$, Bob holds an $n$-bit string $S_B \in \{0,1\}^n$, and they want to decide whether $S_A = S_B$.

**Protocol**

1. Let $S$ be the set of $n^2$ smallest prime numbers.
2. Alice samples a number $p$ from $S$ uniformly at random.
3. Alice sends $p$ and $S_A \mod p$ to Bob (thus, only $O(\log p) \subseteq O(\log n)$ bits are sent, see Q3).
4. After receiving Alice's message, Bob calculates $S_B \mod p$.
5. If $S_A \mod p = S_B \mod p$, Bob decides that $S_A = S_B$, otherwise Bob decides that $S_A \neq S_B$.

# Question 2

Show that this (randomized) communication protocol is correct with a probability of at least $1 - \frac{1}{n}$.

# Question 2

Show that this (randomized) communication protocol is correct with a probability of at least $1 - \frac{1}{n}$.

**Answer**

We consider:

- **Case:** $S_A = S_B$
  Bob always decides $S_A = S_B$ since $S_A \mod p = S_B \mod p$ for all $p$, ensuring correctness with probability $1$.
- **Case:** $S_A \neq S_B$
  Incorrect only if $S_A \mod p = S_B \mod p$, meaning $p$ is a divisor of $|S_A - S_B|$.
  - Since $|S_A - S_B| \leq 2^n$, it has at most $n$ prime factors (Proof).
  - The probability that a random $p \in S$ divides $|S_A - S_B|$ is at most $\frac{n}{|S|} \leq \frac{1}{n}$.

  Correct with probability at least $1 - 1/n$.

Combining both cases, the algorithm is correct with probability at least $1 - 1/n$.

# Question 3 [P2]

Show that any deterministic algorithm solving the equality testing problem requires communicating $\Omega(n)$ bits in the worst case.

# Question 3 [P2]

Show that any deterministic algorithm solving the equality testing problem requires communicating $\Omega(n)$ bits in the worst case.

**Answer**

In the one-way communication model, Alice sends a message to Bob, who then decides whether $S_A = S_B$. A lower bound of $n$ bits follows:

- There are $2^n$ possible $n$-bit strings $S_A$.
- Alice can send at most $2^{n-1}$ distinct $(n-1)$-bit messages.

By the **pigeonhole principle**, at least two distinct $n$-bit strings, say $X$ and $Y$, must map to the same message. This means Bob **cannot distinguish** between:

- $(S_A = X, S_B = X)$
- $(S_A = Y, S_B = X)$

Since Bob must fail in at least one case, at least $n$ bits are required for correctness.

# Random Partition of a Graph

Given a graph $G = (V, E)$ (without self-loops), partition its vertex set into two parts $V = V_1 \cup V_2$ randomly as follows:

- Each vertex $v \in V$ flips an unbiased coin independently:
  - If **heads** (probability $1/2$), $v$ joins $V_1$.
  - If **tails** (probability $1/2$), $v$ joins $V_2$.

## Question 4 [G]/[P3]

Show that the expected number of edges crossing $V_1$ and $V_2$ is exactly $|E|/2$.

[3] Takes the value 1 if a specific event occurs and 0 otherwise.

# Question 4 [G]/[P3]

Show that the expected number of edges crossing $V_1$ and $V_2$ is exactly $|E|/2$.

**Answer**

For each edge $e \in E$, let $X_e$ be an indicator[3] random variable for the event that $e$ crosses between $V_1$ and $V_2$.

**Compute $\mathbf{E}[X_e]$**

For any edge $e = \{u, v\}$, we consider the possible assignments:

- $u, v \in V_1$: Probability $1/4$ (no crossing).
- $u \in V_1, v \in V_2$: Probability $1/4$ (crossing).
- $u \in V_2, v \in V_1$: Probability $1/4$ (crossing).
- $u, v \in V_2$: Probability $1/4$ (no crossing).

Since $e$ crosses with probability $1/4 + 1/4 = 1/2$, we have:

$$\mathbf{E}[X_e] = 1 \cdot \mathbf{Pr}[e \text{ crosses } V_1 \text{ and } V_2] = 1/2.$$

[3] Takes the value 1 if a specific event occurs and 0 otherwise.

Expected[4] number of edges crossing $V_1$ and $V_2$ is:

$$\begin{aligned}
\mathbf{E}[X] &= \mathbf{E}[\sum_{e \in E} X_e] && (X = \sum_{e \in E} X_e) \\
&= \sum_{e \in E} \mathbf{E}[X_e] && \text{(Linearity of expectation)} \\
&= \sum_{e \in E} \frac{1}{2} && \text{(Each edge } \mathbf{E}[X_e] = 1/2) \\
&= \frac{|E|}{2}. && \text{(Sum over all edges)}
\end{aligned}$$

[4]The symbol $\mathbb{E}$ is commonly used as well.

# Question 5 [G]

Is it possible to improve the bound $|E|/2$ in the above result?

If you claim it is possible, propose some ideas and analyze the new bound.

**Answer**

**Assuming $E \neq \emptyset$**

The partitioning algorithm can be slightly improved:

- Pick an edge $e^* = \{u^*, v^*\}$.
- Force $u^* \in V_1$ and $v^* \in V_2$.
- Assign the remaining vertices $V \setminus \{u^*, v^*\}$ randomly.

**Analysis**

$$\mathbf{E}[X_e] = \begin{cases} 1, & \text{if } e = e^*, \\ \frac{1}{2}, & \text{if } e \in E \setminus \{e^*\}. \end{cases}$$

Thus, the expected number of crossing edges is:

$$\mathbf{E}[X] = \sum_{e \in E} \mathbf{E}[X_e] = 1 + \frac{|E| - 1}{2} = \frac{|E| + 1}{2},$$

which is slightly better than $\frac{|E|}{2}$.

**Assuming Even $|V|$**

We can use the following:

- Select $V_1$ uniformly at random from all $(|V|/2)$-vertex subsets of $V$
- Set $V_2 = V \setminus V_1$.

**Analysis**

To compute probability of $e = \{u, v\}$ crosses $V_1$ and $V_2$, we begin with $\Pr(u \in V_1)$:

$$\mathbf{Pr}(u \in V_1) = \frac{|V|/2}{|V|} = \frac{1}{2} \quad \text{(Each vertex is equally likely to be in } V_1\text{)}$$

Given that $u \in V_1$, the probability that $v \in V_2$ is:

$$\begin{aligned} \mathbf{Pr}(v \in V_2 \mid u \in V_1) &= 1 - \mathbf{Pr}(v \in V_1 \mid u \in V_1) && \text{(Complement rule)} \\ &= 1 - \frac{|V|/2 - 1}{|V| - 1} = \frac{|V|/2}{|V| - 1}. && \text{(Probability of } v \text{ being in } V_1\text{)} \end{aligned}$$

Similarly, for $u \in V_2$, the case is symmetric; we sum both cases:

$$\begin{aligned}
\mathbf{E}[X_e] &= \mathbf{Pr}(u \in V_1, v \in V_2) + \mathbf{Pr}(u \in V_2, v \in V_1) \\
&= 2 \cdot \mathbf{Pr}(u \in V_1, v \in V_2) && \text{(Two symmetric cases)} \\
&= 2 \cdot \mathbf{Pr}(u \in V_1) \cdot \mathbf{Pr}(v \in V_2 \mid u \in V_1) && \text{(conditional probability)} \\
&= 2 \times \left( \frac{1}{2} \times \frac{|V|/2}{|V|-1} \right) = \frac{|V|/2}{|V|-1}.
\end{aligned}$$

By linearity of expectation, summing over all edges:

$$\mathbf{E}[X] = \sum_{e \in E} \mathbf{E}[X_e] = |E| \times \frac{|V|/2}{|V|-1} = \frac{|E|}{2} \cdot \frac{|V|}{|V|-1}.$$

**Remarks**: This bound is **tight** for complete graphs, and a bound that is tight for complete graphs can also be obtained similarly for odd $|V|$.

# Practical [Optional]

Practical repo: To help you further your understanding, not compulsory; Work for Snack!

1. Implement `freivalds`.
2. Check that you get this output:

```
('Test 1', True)
('Test 2', True)
('Test 3', True)
('Test 4', True)
```